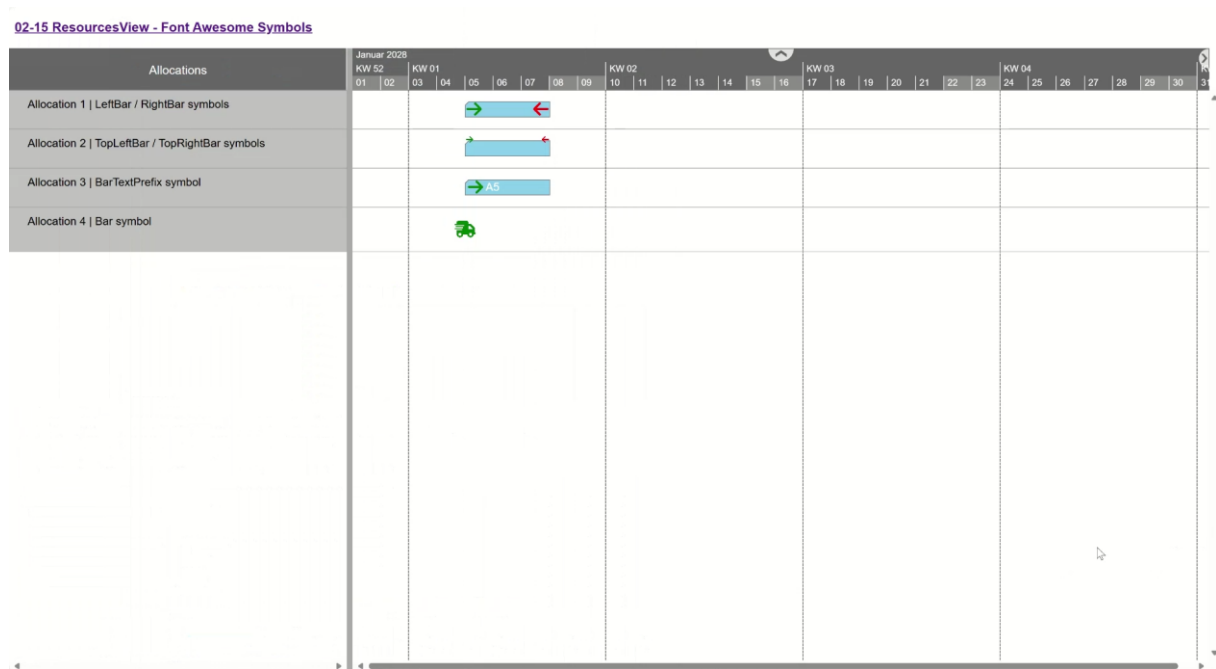


21-03 Resources view – Visualization – Font Awesome symbols

This example shows how to include font awesome symbols so that they can be attached to the bars. Font Awesome is a widely used icon library and toolkit that is mainly used in web design and web development. It contains a variety of scalable vector icons and social logos. There are free and paid symbols.



In order to access the Font Awesome symbols, two scripts must be included:

```
<script src="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.7.2/js/solid.min.js">
</script>
<script src="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.7.2/js/fontawesome.min.js"
  data-observe-mutations="false"></script>
```

Font Awesome traditionally uses `<i>` tags to display icons. In newer versions (from Font Awesome 5), this is replaced internally by SVG if the SVG+JS implementation is used.

```
<i data-fa-symbol="arrow-right" class="fas fa-arrow-right fa-fw"></i>
<i data-fa-symbol="arrow-left" class="fas fa-arrow-left fa-fw"></i>
<i data-fa-symbol="truck-fast" class="fas fa-truck-fast fa-fw"></i>
```

The meaning for class attribute is as follows:

`class="fas fa-arrow-right fa-fw"`

- `fas`: Activates the "Font Awesome Solid" style (solid icons).
- `fa-arrow-right`: The concrete icon - here the right arrow.
- `fa-fw`: Bedeutet "fixed width" – all icons have the same width, which is useful for lists or menus.

`data-fa-symbol="arrow-right"`

This is the crucial point for icon usage. This attribute tells Font Awesome that this icon should be defined as an icon under the alias name **arrow-right**. This means:

- The icon is not displayed directly.

- It is stored in a hidden icon definition block (similar to <symbol> in SVG).

The Visual Scheduling Widget symbol definition refers to the alias name. It is important to use the prefix # in the URL. The attribute **InclusionMode** ensures that the URL is interpreted appropriately.

```
const symbol1 = {ID: "Symbol1", Color: "green", URL: "#arrow-right", InclusionMode:
    Enum.SymbolInclusionMode.EmbeddingReference};
```

In this context, it is important to encapsulate the code in a function:

```
document.addEventListener("DOMContentLoaded", function () { ... })
```

The DOMContentLoaded event is triggered as soon as the HTML document is fully loaded and parsed, before images, stylesheets or other resources have finished loading.

In the SVG+JS mode of Font Awesome, SVG symbols are generated by JavaScript. At runtime, certain HTML tags such as <i class="fas fa-arrow-right"> are replaced by embedded <svg> elements. This conversion is done by a JavaScript that Font Awesome itself provides.

Font Awesome can only registers the symbols internally after the DOM has been loaded and its own script has run. If an attempt is made to access symbols beforehand, they will not be replaced correctly.

Further use of the symbols corresponds to the example:

02-14_ResourceView_AllocationSymbols.pdf

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>02-15 NETRONIC VSW SE</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <link href="../../VSWidget/nwaf-apptools.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-table.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-gantt.min.css" rel="stylesheet"/>
  <link href="../../VSWidget/nwaf-rab.min.css" rel="stylesheet"/>
  <script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8/hammer.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/d3@7.9.0/dist/d3.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/tinycolor2@1.6.0/cjs/tinycolor.min.js"></script>
  <script src="../../VSWidget/nwaf-apptools.min.js"></script>
  <script src="../../VSWidget/nwaf-table.min.js"></script>
  <script src="../../VSWidget/nwaf-gantt.min.js"></script>
  <script src="../../VSWidget/nwaf-rab.min.js"></script>
  <script src="../../LicenseKey.js"></script>
  <script src="Miscellaneous.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.7.2/js/solid.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.7.2/js/fontawesome.min.js"
    data-observe-mutations="false"></script>
</head>
<body>
  <div id="toolbar">
    <h3>
      <a href="docs/21-03_ResourceView-Visualization-Font_Awesome_symbols.pdf"
        target="_blank">21-03 Resources view - Visualization - Font Awesome symbols</a>
    </h3>
  </div>
  <div id="VSWidget"></div>
  <i data-fa-symbol="arrow-right" class="fas fa-arrow-right fa-fw"></i>
  <i data-fa-symbol="arrow-left" class="fas fa-arrow-left fa-fw"></i>
  <i data-fa-symbol="truck-fast" class="fas fa-truck-fast fa-fw"></i>
  <script>
    Miscellaneous.ready(() => {
      const { BarShape, BarDragModes, RowDesigns,
        SymbolInclusionMode, ViewType } = netronic.nVSW;
```

```

// run i2svg after loading DOM content and then create and configure VSW
window.FontAwesome.dom.i2svg().then(() => {
  const timeAreaStart = new Date(2028, 0, 1);
  const timeAreaEnd = new Date(2028, 1, 1);

  const options = {
    licenseKey: window.VSW_LICENSE_KEY,

    viewType: ViewType.Resources,

    start: timeAreaStart,
    end: timeAreaEnd,

    defaultResourceExpandedRowDesign: RowDesigns.Bars,
    defaultResourceCollapsedRowDesign: RowDesigns.Bars |
      RowDesigns.BarsInHiddenDescendantRows |
      RowDesigns.BarsStacked,

    defaultAllocationAllowedBarDragModes: BarDragModes.None,
    defaultAllocationBarSelectable: false,
    defaultResourceRowSelectable: false,
    multipleSelectionEnabled: false,
    defaultResourceTableRowDefinitionID: "CustomRowDef",
    ignoreCalendarOnAllocationBarInteractions: true,
    topRowMarginInTimeArea: 17,
    bottomRowMarginInTimeArea: 17,
    fixedTableColumnWidth: 0,
  };

  const vsWidget = Miscellaneous.instantiateVSWidget(document.querySelector("#VSWidget"), options);

  const symbols = createSymbols();
  const tableRowDefinitions = createTableRowDefinitions();
  const resources = createResources();
  const allocations = createAllocations();

  vsWidget.addSymbols(symbols);
  vsWidget.addTableRowDefinitions(tableRowDefinitions);
  vsWidget.addResources(resources);
  vsWidget.addAllocations(allocations);
  vsWidget.render();

  function createSymbols() {
    const symbols = [
      {
        ID: "Symbol1",
        Color: "green",
        URL: "#arrow-right",
        InclusionMode: SymbolInclusionMode.EmbeddingReference,
      },
      {
        ID: "Symbol2",
        Color: "red",
        URL: "#arrow-left",
        InclusionMode: SymbolInclusionMode.EmbeddingReference,
      },
      {
        ID: "Symbol3",
        Color: "green",
        URL: "#truck-fast",
        InclusionMode: SymbolInclusionMode.EmbeddingReference,
      }
    ];

    return symbols;
  }

  function createTableRowDefinitions() {
    const tableRowDefinitions = [
      {
        ID: "CustomRowDef",
        CellDefinitions: [
          {

```

```

        TitleText: "Allocations",
        Width: 480,
    },
],
},
];

return tableRowDefinitions;
}

function createResources() {
    const total = 4;
    const resources = [];
    for (let i = 1; i <= total; i++) {
        const resource = {
            ID: `Res${i}`,
        };
        resources.push(resource);
    }
    resources[0].TableText = "Allocation 1 | LeftBar / RightBar symbols";
    resources[1].TableText = "Allocation 2 | TopLeftBar / TopRightBar symbols";
    resources[2].TableText = "Allocation 3 | BarTextPrefix symbol";
    resources[3].TableText = "Allocation 4 | Bar symbol";

    return resources;
}

function createAllocations() {
    const allocations = [
        {
            ID: "Alloc1",
            ResourceID: "Res1",
            Entries: [
                {
                    Start: new Date(2028, 0, 5),
                    End: new Date(2028, 0, 8),
                    Color: "skyblue",
                },
            ],
            LeftBarSymbolID: "Symbol1",
            LeftBarSymbolHeight: 25,
            LeftBarSymbolWidth: 25,
            RightBarSymbolID: "Symbol2",
            RightBarSymbolHeight: 25,
            RightBarSymbolWidth: 25,
        },
        {
            ID: "Alloc2",
            ResourceID: "Res2",
            Entries: [
                {
                    Start: new Date(2028, 0, 5),
                    End: new Date(2028, 0, 8),
                    Color: "skyblue",
                },
            ],
            TopLeftBarSymbolID: "Symbol1",
            TopRightBarSymbolID: "Symbol2",
        },
        {
            ID: "Alloc3",
            ResourceID: "Res3",
            Entries: [
                {
                    Start: new Date(2028, 0, 5),
                    End: new Date(2028, 0, 8),
                    Color: "skyblue",
                },
            ],
            BarText: "A5",
            BarTextPrefixSymbolID: "Symbol1",
            BarTextPrefixSymbolHeight: 25,
            BarTextPrefixSymbolWidth: 25,
        },
    ],
}

```

```

        {
            ID: "Alloc4",
            ResourceID: "Res4",
            Entries: [
                {
                    Start: new Date(2028, 0, 5),
                    End: new Date(2028, 0, 8),
                },
            ],
            BarShape: BarShape.Symbol,
            BarShapeSymbolID: "Symbol3",
            BarHeight: 30,
            BarShapeSymbolWidth: 30,
        },
    ];

    return allocations;
}
});
</script>
</body>
</html>

```